**입력값의 정규화**

**신경망의 훈련을 빠르게 할 수 있는 하나의 기법**은 **입력을 정규화**하는 것이다. 두 개의 입력 특성이 있는 훈련세트가 있다. 입력특성 x가 2차원이다. 그리고 훈련세트의 산포도가 있다. **두 단계**에 따라서 **입력을 정규화**하겠다.

텍스트, 영수증, 폰트, 화이트이(가) 표시된 사진

자동 생성된 설명

**첫번째**는 **평균을 빼는 것**이다. 즉, 0으로 만든다. 0의 평균을 갖게 될 때까지 훈련세트를 이동한다.

**두번째**는 **분산을 정규화**하는 것이다. 특성 x1이 특성 x2보다 더 큰 분산을 갖고 있다. (\*\*는 요소별 제곱을 나타냄, σ​2​​는 각 특성의 분산에 대한 벡터, x​(i)^2는 분산) x1과 x2 분산은 이제 모두 x1과 같다. 두 단계를 그래프로 나타내면 다음과 같다.

라인, 도표, 그래프, 폰트이(가) 표시된 사진

자동 생성된 설명

추가적으로 이것을 훈련 데이터를 확대하는데 사용한다면 **테스트 세트를 정규화할 때**도 **훈련 데이터에 사용한 μ와 σ를사용해라**(훈련세트와 테스트세트를 다르게 정규화하고 싶지는 않을 것이다.) μ와 σ의 값이 어떤 값이든 위 두가지 식에 사용하면 정확히 같은 방식으로 테스트 세트를 확장하게 된다.

훈련 세트와 테스트 세트에서 별개로 μ와 σ^2를 추정하는 대신에 말이다. 왜냐하면 훈련 샘플과 테스트 샘플 모두 훈련 데이터에서 계산된 것과 같은 μ와 σ^2에 의해 정의된 변형을 거치기를 원하기 때문이다.

 그럼 왜 입력특성을 정규화하기를 원할까? 비용함수의 정의는 다음과 같다. 정규화되지 않은 입력특성을 사용하면 비용함수가 매우 구부러진 활처럼 가늘고 긴 모양의 비용함수가 된다.

폰트, 화이트, 텍스트, 친필이(가) 표시된 사진

자동 생성된 설명

**i) 특성을 정규화 하지 않은 경우**

그림이(가) 표시된 사진

자동 생성된 설명라인이(가) 표시된 사진

자동 생성된 설명

우리가 찾으려고 하는 최솟값은 오른쪽 그림의 화살표에 위치한다. 만약 특성들이 매우 다른 크기를 갖는다면(예를들어 x1이 1~1000, x2가 0~1) 매개변수에 대한 비율, 값의 범위는 w1과 w2가 아주 다른 값을 갖는다.

**ii) 특성을 정규화한 경우**

다채로움, 원, 무지개, 스크린샷이(가) 표시된 사진

자동 생성된 설명

 특성을 정규화하면 비용함수는 평균적으로**대칭적인 모양**을 갖게 된다.

**i)의 비용함수에 경사하강법**을 실행한다면 **매우 작은 학습률**을 사용하게 된다. 그 이유는 i)의 그래프에서 경사하강법은최종적으로 최솟값에 이르는 길을 찾기위해 앞뒤로 왔다 갔다 하기위해 많은 단계가 필요하기 때문이다. 반면에 **ii)의 비용함수에 경사하강법**을 실행한다면 **어디서 시작하든 최소값으로 바로** 갈 수 있다. i)과 달리 큰 스텝으로 전진이 가능하기 때문이다.

라인, 도표, 원이(가) 표시된 사진

자동 생성된 설명

따라서 특성이 비슷한 크기를 가질 때 비용함수가 더 둥글고 최적화하기 쉬운 모습이 된다는 대략적인 직관을 얻을 수 있다. 1~1000, 0~1이 아닌 대부분 -1~1 혹은 서로 비슷한 분산으로 비용함수 J를 최적화하기 쉽고 빠르게 만든다. 너무 다른 범위는 최적화 알고리즘에 방해가 된다.

**모든 것을 0의 평균으로 설정하고 모든 특성을 비슷한 크기로 보장할 수 있는 분산을 설정하면 학습 알고리즘이 빠르게 실행되는 것을 도울 것이다.**

**경사소실/경사폭발**

 매우 깊은 신경망을 훈련시킬 때 나타나는 문제점은 경사의 소실과 폭발이다. 즉, 미분값 혹은 기울기가 아주 작아지거나 커질 수 있다. 이는 훈련을 어렵게 만든다. 경사 폭발과 소실의 문제점이 무엇인지 알아보고 무작위의 가중치 초기화에 대한 신중한 선택으로 문제를 상당히 줄이는 방법을 다뤄보자.

그림처럼 두개의 은닉 유닛만을 가지는 매우 깊은 신경망을 훈련시키는 경우를 생각해보자. 간단함을 위해 활성화함수 g(z)가 선형 활성화 함수를 사용한다고 하자. b^[l]=0 이라고 가정한다.

라인, 폰트, 타이포그래피이(가) 표시된 사진

자동 생성된 설명

 이 경우에 출력 y = w^[l] \* w^[l-1] \* w^[l-2] \* ... \* w^[3] \* w^[2] \* w^[1] \* x 가 된다. 수학적으로 증명하다보면 이 모든 행렬들의 곱은 y의 예측값이 된다.

친필, 텍스트, 스케치, 그림이(가) 표시된 사진

자동 생성된 설명y 수식의 수학적 증명 과정텍스트, 폰트, 화이트, 영수증이(가) 표시된 사진

자동 생성된 설명

경사하강법에서 계산하는 경사가 층의 개수에 대한 함수로 기하급수적으로 증가하거나 감소한다는 것을 보여주는데 사용할 수도 있다. 현대 신경망은 보통 L이 150의 값을 갖는다. 마이크로소프트는 최근 852개의 층을 가진 신경망을 결과로 내놓았다. 그러나 이런 깊은 신경망에서**활성값이나 경사가 L에 대한 함수로 기하급수적으로 증가하거나 감소한다면 값들은 아주 커지거나 작아질 수 있다.** 특히 경사가 기하급수적으로 작은 경우에는 더욱 **훈련시키는 것이 어려워진다. 경사하강법은 아주 작은 단계만을 진행하고 학습시키는데 아주 오랜 시간이 걸리게 된다.**

**심층 신경망의 가중치 초기화**

깊은 신경망에서 경사 소실/폭발은 문제가 있다. 이를 해결하는 방법 중 하나는 **신경망에 대한 무작위의 초기화를 더 신중하게 택하는 것**이다.

먼저 단일 뉴런에 대한 가중치를 초기화하는 예제를 살펴보자. 그리고 더 깊은 네트워크로 이를 일반화시킬 것이다.

폰트, 라인, 도표, 원이(가) 표시된 사진

자동 생성된 설명단일 뉴런

나중에 다룰 더 깊은 망에서 입력은 a^[l]인 어떤 층이 될 것이다. z=w\_1x\_1 + w\_2x\_2 + x\_nx\_n의 값을 갖는다. b=0이라 한다(무시). z의 값이 너무 크거나 작아지지 않도록 만들어야 한다. n의 값이 클수록 w\_i의 값이 작아져야 한다. (z는 w\_ix\_i의 합이기 때문에 이를 다 더하면 각가의 항이 작아지기를 바라기 때문이다.)

* **w\_i의 분산을 1/n으로 설정한다.** (n: 입력 특성의 개수) 따라서 특정 층에 대한 가중치 행렬 w^[l]은 아래 코드와 같다.
  + : w^[l] = np.random.rand(shape)\*sp.sqrt(1/n^[l-1])
* **ReLU 활성화 함수를 사용하는 경우, w\_i의 분산을 2/n^[l-1]로 설정한다.** g^[l](z) = ReLU(z)이다. 따라서 입력특성 혹은 활성값의 평균이 대략 0이고 표준편차 1을 갖는다면, 이 역시 비슷한 크기를 갖는다. 완전히 해결하지는 못하지만 경사소실과 폭발문제에 도움을 줄 수 있다.

w^[l] = np.random.rand(shape)\*sp.sqrt(2/n^[l-1])

* tanh 활성화 함수를 사용하는 경우, w\_i의 분산을 1/n^[l-1] 또는 2/n^[l-1]+n[l] 으로 설정한다.

w^[l] = np.random.rand(shape)\*sp.sqrt(1/n^[l-1])

w^[l] = np.random.rand(shape)\*sp.sqrt(2/n^[l-1]+n^[l])

실제로 위의 식들은 그저 시작점을 제공한다. 즉, **가중치 행렬의 초기화 분산에 대한 기본값을 줄 뿐**이다. 이와 같은 분산을 원한다면 분산 매개변수는 하이퍼파라미터로 조정할 또 다른 값이 된다. 따라서 이 식에 곱할 매개변수를 가질 수 있고 하이퍼파라미터 탐색의 일부로 그 곱하는 수를 조장할 수 있다. 가끔 하이퍼파라미터를 조정하는 것은 어느정도의 효과를 갖는다.

이를 통해 경사소실과 폭발 문제에 대한 직관을 얻는데 도움이 됐길 바란다. 가중치를 초기화하는 합당한 크기를 선택하는데에도 말이다. 이를 통해 가중치가 너무 빨리 폭발하거나 0으로 수렴하지 않기를 바란다. 가중치 혹은 경사의 폭발과 소실 없이 상당히 깊은 네트워크를 훈련시킬 수 있다. 이상으로 **깊은 네트워크를 훈련시킬 때 신경망을 더 빨리 훈련시키는 또다른 기법이었다.**

**기울기의 수치 근사**

역전파를 구할때 **경사 검사**라는 테스트가 있다. **경사 검사는 역전파를 알맞게 구현했는지 확인하는데 이용**한다. 왜냐하면 모든 수식을 작성해도 세부사항이 올바른지 100% 확신할 수 없기 때문이다. 따라서 경사 검사를 구현하기 위해 경사의 계산을 수치적으로 근사하는 방법을 알아보자. 아래는 f(θ)=θ^3의 그래프이다. θ가 1인 경우이다. ε은 아주 작은 수이다.

라인, 도표, 경사, 그림이(가) 표시된 사진

자동 생성된 설명라인, 그림, 도표, 아동 미술이(가) 표시된 사진

자동 생성된 설명

f가 *θ-ε*인 점과 *θ+ε*인 점에서 높이/너비를 더 큰 삼각형에서 계산한다면 더 나은 경사비율을 얻을 수 있다(오른쪽 그림 참고). 왜냐하면 더 큰 삼각형에서 높이/너비를 구하는 것이 *θ*에서 도함수를 근사하는데 더 나은 값을 구할 수 있기 때문이다.  (큰 초록 삼각형의 높이) = f(θ+ε) - f(θ-ε), (오른쪽 그림의 초록 삼각형의 너비) = 2ε

 따라서 (f(θ+ε) - f(θ-ε))/ 2는 g(θ)와 비슷해야한다. 양쪽의 차이를 이용하는 방법을 사용하면 한쪽을 이용해 구하는 것보다 더 가까운 값이 나옴을 알 수 있다. 이것을 역전파의 경사 검사에서 사용한다면 한 쪽 차이만 사용하는 것보다 두배는 느리게 실행된다.

텍스트, 친필, 폰트, 서예이(가) 표시된 사진

자동 생성된 설명

**경사 검사**

경사검사는 시간을 절약하고 역전파의 구현에 대한 버그를 찾는데 도움을 준다. 어떻게 하는지 알아보자.

**Take W^[1], b^[1], .... ,W^[L], b^[L] and reshape into a big vector θ.**(신경망은 매개변수 W^[1], b^[1]부터 W^[L], b^[L]까지 갖는다.) 첫번째는 모델 안에 있는 매개변수들을 하나의 큰 벡터 θ로 바꾸는 것이다. 행렬 W^[1]을 벡터로 크기를 바꾼다. 모든 W 행렬을 받아서 벡터로 바꾸고 모두 연결시킨다. 그 결과 매우 큰 벡터 매개변수 θ를 얻게 된다. 그러면 비용함수 J(W, b)는 J(θ)로 변한다.

**Take dW^[1], db^[1], .... ,dW^[L], db^[L] and reshape into a big vector dθ.**dW^[1], db^[1], ...의 매개변수를 매우 큰 벡터 dθ로 만든다. dW^[1]을 벡터로 바꾸고 db^[1]은 이미 벡터이다. 모든 dW는 행렬이다. dW^[1]은 W^[1]과 같은 차원이고 db^[1]은 b^[1]과 같은 차원이다. 같은 방식으로 크기를 바꾸고 연결해 모든 미분값을 매우 큰 벡터 dθ로 바꿀 수 있다.

**그렇다면, dθ가 비용함수 J(θ)의 기울기일까?**경사 검사의 구현방법을 보자.

J는 이제 매우 큰 매개변수 θ에 관한 함수이다. J(θ) = J(θ1, θ2, θ3, ...) 경사 감소를 위해 반복문을 구현한다. θ요소 각각에 대해 dθapprox[i]를 계산해보자.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

dθapprox[i]은 근사적으로 dθ[i]와 같아야 한다. 이는 함수 J의 θ\_i에 대한 편미분과 같다. dθapprox와 dθ가 같은 차원이 된다. 이 두 벡터가 근사적으로 같은지 확인해야한다. 두 벡터의 유사도는 유클리드 거리를 계산하면 알 수 있다. dθapprox - dθ의 L2 노름을 구한다. (제곱하지 않음에 주의) 벡터의 길이로 정규화하기 위해 ||dθapprox|| + ||dθ||의 유클리드 길이로 나눠준다. 보통 수식의 값이 10^(-7)보다 작으면 근사가 매우 잘 되었음을 의미한다.

따라서 신경망의 정방향 전파 또는 역전파를 구현할 때 경사검사에서 상대적으로 큰 값이 나온다면 (10^(-3)) 버그의 가능성을 의심해야 한다. 디버깅의 과정을 거친 후 경사검사에서 작은 값이 나온다면 구현이 잘 된 것이다.

**경사 검사 시 주의할 점**

**신경망에서 경사 검사를 구현**하기 위한 실질적인 방법과 팁을 알아보자.

* **Don't use in training - only to debug** : 모든 i의 값에 대해 dθapprox[i]를 계산하는 것은 느리기 때문에 훈련에서 사용하지 않고 디버깅을 위해서만 사용한다.
* **If algorithm fails grad check, look at components to try to identify bug** : 만약 경사 검사의 알고리즘이 실패한다면 개별적인 컴포넌트를 확인해 버그를 확인해라. 즉, dθapprox가 dθ에서 매우 먼 경우 서로 다른 i에 대하여 어떤 dθapprox[i]의 값이 dθ[i]의 값과 매우 다른지 확인할 것이다.
* **Remember regularization** : 경사 검사할때 사용하는 정규화 항을 기억해라. dθ는 θ에 대응하는 J의 정규화 항도 포함하기 때문에 경사 검사 계산시 같이 포함해야 한다.

폰트, 텍스트, 타이포그래피, 화이트이(가) 표시된 사진

자동 생성된 설명

* **Doesn't work with dropout** : 경사 검사는 드롭아웃에서는 작동하지 않는다. 드롭아웃은 모든 반복마다 은닉 유닛의 서로 다른 부분집합을 무작위로 삭제하기 때문에 적용하기 쉽지 않다. 따라서 드롭아웃을 이용한 계산을 이중으로 확인하기 위해 경사검사를 확인하기는 어렵다. (드롭아웃을 끄고 알고리즘이 최소한 드롭아웃 없이 맞는지 확인하고, 다시 드롭아웃을 켠다.)
* **Run at random initialization, perhaps again after some training** : 가끔 일어나는 무작위 초기화(w와 b가 0에 가까울 때 경사하강법의 구현에 맞게된 경우)에서 경사 검사를 실행하는 것이다. 그리고 네트워크를 잠시 동안 훈련해서 w와 b가 0에서 멀어질 수 있는 시간을 준다. 일정 수의 반복을 훈련한 뒤에 경사 검사를 다시 해보는 방법이 있다.

텍스트, 폰트, 친필, 스크린샷이(가) 표시된 사진

자동 생성된 설명